The PC-AT Boot Process and Option ROMs

Roy Wade
Staff Software Engineer
Storage Components Division
LSI Logic Corporation

Ramin Neshati
Developer Guides
Technical Program Manager
Intel Corporation

Table of Contents

(Click on page number to jump to sections)

The PC-AT Boot Process and Option ROMs

Roy Wade
Staff Software Engineer
Storage Components Division
LSI Logic Corporation

Ramin Neshati
Developer Guides
Technical Program Manager
Intel Corporation

Overview

The ubiquitous PC-AT architecture has reached a point where system designers and Independent Hardware Vendors (IHVs) are actively looking for ways to extend its capabilities, performance, ease-of-use, scalability, and other design characteristics. Much has changed since the introduction of the PC in the early 1980s: the system processors, I/O bus, memory map, boot process, graphics, etc., have all undergone drastic improvements. However, we stand today at the cusp of even greater technology transitions in systems design, allowing for vastly improved system performance, compatibility, multi-vendor interoperability, and "headroom" for scalability, future expansion, and growth.

The introduction of the IA-64 processor family allows system designers to make a clean break from the past and introduce new and innovative system designs that will do for server systems what the earlier Intel x86 processors did for the desktop PC—a massive proliferation of high-performance, low-cost server systems and solutions. Attempts at reducing or removing legacy technologies, allowing room for innovation, and extending the ease-of-use, performance, and scalability characteristics of server systems based on the Intel® Architecture have received special attention recently. A group of leading companies in the computer industry has formed working groups to collectively create an evolutionary path for the transition away from legacy technologies. LSI Logic and Intel, both leading companies in the markets they serve, have been at the forefront of paving the way for extending the Intel Architecture into the enterprise and the data center.

The PC-AT boot process lies at the core of this legacy migration trend. The introduction of the Extensible Firmware Interface (EFI), an abstraction interface that allows the de-coupling of the hardware from the operating system boot loader and provides other run-time services, has been enthusiastically adopted by the industry. However, the initial release of the EFI specification has not fully addressed an area that is of special importance to IHVs: option ROMs. However, efforts are already afoot to address this issue in the next release of the EFI specification. Quite simply, an option ROM allows the IHV to supply additional firmware to boot and configure its peripheral device and to extend the functionality of the system firmware (BIOS) during the boot process. There is no shortage of documentation detailing the PC-AT boot sequence. As PC technology has evolved, the number of specifications covering this topic has grown resulting in a fragmented image that takes some effort to grasp.

This article provides a comprehensive discussion of the PC-AT boot process in terms of today's technologies and architectures. Specifically discussed are PCI expansion ROMs for Plug and Play (PnP) devices and the details of their role in the boot process. A companion white paper that describes a legacy-reduced system boot process is currently in the early stages of formulation. We will defer the discussion of EFI option ROMs to this near-future effort.

Option ROMs Defined

The following is a general description of the current IA-32 option ROM architecture. Only Plug and Play compatible option ROMs are discussed. For a comprehensive and historical perspective, refer to the Plug and Play BIOS specification.

As was mentioned, an option ROM is a firmware component associated with a specific add-on device. The associated device may reside on the system-board or on a Plug and Play card. Accordingly, a device's option ROM will reside in system ROM or in PnP card ROM. Option ROMs are intended to isolate a hardware device by providing an abstract interface that implement device-specific functions, including:

- Power-on self-test
- Initialization
- Interrupt service routines
- BIOS routines

The I/O services provided by an option ROM serve as a translation layer between differing protocols. For example, an option ROM may provide a legacy Int 10H video interface to an AGP device; or provide a legacy Int 13h disk drive interface to one or more disk drives attached to a SCSI bus. The PnP option ROM model is generic and can support any Plug and Play device. Requirements specific to an application are detailed in existing specifications and are beyond the scope of this article (e.g., a PCI-to-USB PnP product would be concerned with the appropriate PCI and USB specifications).

Typically a PnP device will provide at least one option ROM image, but some architectures allow for more (e.g., PCI). It is not required that a PnP device provides an associated option ROM, hence they are optional. Option ROMs facilitate the Plug and Play concept by adhering to a common image format, which indicates flexible resource requirements in support of automatic configuration.

During the boot process, option ROM images are copied from their ROM to main memory (RAM) and their ROM address range is mapped (shadowed) to main memory. This is done to increase execution speed by taking advantage of the faster access time of RAM versus ROM. The system BIOS is responsible for this expansion process during boot. The system BIOS loads and initializes option ROM images at its discretion. In newer systems, only those option ROMs required to boot the operating system are loaded. The operating system then loads its own drivers for all PnP devices while shadowed option ROMs remain resident (and dormant) in RAM.

The primary purpose of the pre-boot subsystem is to provide a set of services to find and execute an operating system loader in a reliable and expeditious manner. The pre-boot environment can also provide error checking, error logging/reporting and error recovery as well as support system expansion.

Two chief components in the boot process are the system BIOS and Initial Program Load (IPL) device option ROMs. The system BIOS is a single instance entity that controls the sequence of boot events. It provides abstracted services, enumerates hardware devices, loads option ROMs, and performs resource management. The system BIOS gains control of the initial boot process by intercepting a power on or reset. It invokes the system bootstrap loader to load the operating system (possibly via an option ROM) from the IPL device and execute it. As discussed previously, option ROMs provide services to access the devices attached to their bus, in this case the boot device.

PC-AT Boot Model

The following discussion of the PC-AT boot model assumes that the reader is familiar with:

PnP compatibility for all components (system BIOS, devices, option ROMs, operating system)
PCI 2.1 compliance
BIOS Boot Specification support for system BIOS and IPL device option ROMs
Device Driver Initialization Model (DDIM) as specified in the PnP BIOS Specification

Figure 1 below shows an example system configuration that may prove helpful in a description of the boot sequence.

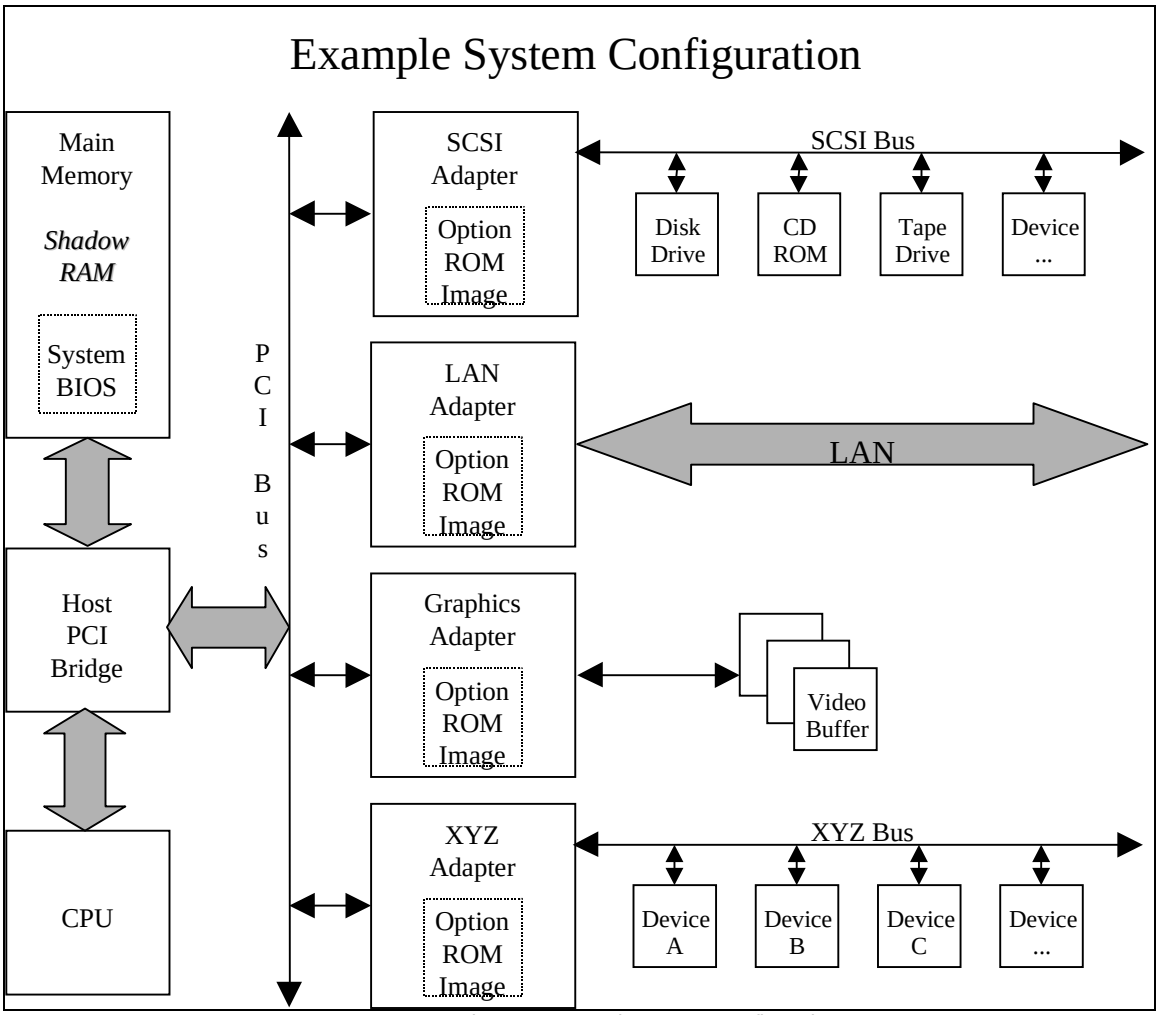

Figure 1 - Example System Configuration

The BIOS Boot Specification defines a method to specify the IPL devices to attempt to boot from and prioritize their order. This method requires that option ROMs identify their IPL devices to the system BIOS and that the system BIOS provides a mechanism for the user to control the IPL order. The comprehensive description of this mechanism can be found in the BIOS Boot (BBS) Specification and is not repeated here.

IPL Sequence of Events

Below is the sequence of events to define the IPL order. In this example, it is assumed that at least one IPL candidate device does exist.

IPL Sequence of Events:

- The system BIOS captures control of the system after power on or reset.
- The system BIOS performs the following functions:
- Initializes the primary output device (display).
- Initializes the primary input device (keyboard).
- Performs self-test functions (RAM test etc.).
- The system BIOS scans the system's busses (e.g., PCI) for PnP option ROM Headers.
- The system BIOS performs resource conflict resolution and allocation. The algorithm used in the resource allocation process is beyond the scope of this specification. In the case of PCI, the resource requirements (IRQs, I/O, memory, etc.) for an option ROM are specified in the Configuration Registers.
- The system BIOS displays a prompt that the operator uses to enter the Setup Utility.
- The system BIOS reads previous settings from non-volatile memory.
- The system BIOS scans the system's busses to detect PnP expansion headers for option ROMs

that provide access to potential IPL devices. For each IPL device option ROM that is found:
- The system BIOS copies the ROM image into main memory.
- The system BIOS calls the image's initialization entry point, which is specified in the PnP option ROM Header. The option ROM performs the following:
- Performs required initialization, and determines the device(s) it is to control.
- If there are no devices to support, the option ROM "resizes" itself to zero in on its option ROM Header, then returns from its initialization entry point.
- If there are devices to support, the option ROM creates a linked list with a new PnP expansion header for each device found. The PnP expansion header fields will uniquely identify each device. A field of particular interest to the user will be the Product Name String. The option ROM then returns from its initialization entry point.
- At this point all option ROMs with PnP expansion headers have been initialized. Option ROMs with device(s) to support will not have hooked any interrupts (e.g., Int 13h).
- The system BIOS now scans main memory to detect all PnP expansion headers for potential IPL devices. For each PnP expansion headers that is found:
- If a BEV header, the entry is added to the IPL priority list.
- If a BCV header, the entry is added to the BCV priority list.

NOTE: See [BIOS Boot Specification Version 1.01] for the distinction between the IPL and BCV priority lists.
- The system BIOS presents the IPL and BCV priority lists to the user via a Setup menu.
- The user manipulates the IPL and BCV priority lists as desired, saves the settings, then exits Setup.
- The system BIOS saves the settings to non-volatile memory and automatically reboots the system.

PnP BBS System

We are now ready to discuss the sequence of events that occur during the boot process of a PnP BBS system. The focus is directed toward the interaction between the system BIOS and option ROMs.

Plug and Play Bios Boot System Sequence of Events:

- The system BIOS captures control of the system after power on or reset.
- The system BIOS performs the following functions:
- Initializes the primary output device (display).
- Initializes the primary input device (keyboard).
- Performs self-test functions (RAM test, etc.).
- The system BIOS scans the system's busses (e.g., PCI) for PnP option ROM headers.
- The system BIOS performs resource conflict resolution and allocation. The algorithm used in the resource allocation process is beyond the scope of this specification, see [PnP BIOS specification reference]. In the case of PCI, the resource requirements (IRQs, I/O, memory, etc.) for an option ROM are specified in the Configuration Registers.
- The system BIOS displays a prompt that the operator can use to enter the Setup utility. The prompt is ignored in this case.
- The system BIOS scans the systems' busses (e.g., PCI) to detect PnP expansion headers for option ROMs that provide access to potential IPL devices. For each IPL device option ROM that is found:
- The system BIOS copies the ROM image into main memory.
- The system BIOS calls the image's initialization entry point, which is specified in the PnP option ROM header. The option ROM performs the following:
- Initialization. It also determines if the device(s) it is to control.
- If there are no devices to support, the option ROM "resizes" itself to zero in its option ROM header then returns from its initialization entry point.
- If there are devices to support, the option ROM creates a linked list with a new PnP expansion header for each device found. The PnP expansion header fields will uniquely identify each device. The option ROM then returns from its initialization entry point.

- At this point, all option ROMs with PnP expansion headers have been initialized. Option ROMs with device(s) to support will not have hooked any interrupts (e.g., Int 13h).
- The system BIOS reads settings from non-volatile memory, including the IPL and BCV priority lists. If non-volatile memory is corrupt, default IPL and BCV priority lists are created.
- The system BIOS builds the run-time BCV priority list.
- Main memory is scanned to detect all PnP expansion headers that have BCV entry points.
- Each BCV entry that is found is appended to the run-time BCV priority list.
- The system BIOS compares the NVM and run-time BCV priority lists (actually it compares the number of items in the lists). For the purposes of this discussion we assume the lists are the same (if the lists do not match the system BIOS may clear the NVM BCV priority list). Either way, the run-time BCV priority list is used from this point on.
- The system BIOS calls each BCV entry (extracted from the PnP expansion header) in the order specified in the BCV priority list. For each entry in the BCV priority list, the system BIOS calls the associated BCV entry point to request the option ROM to install Int 13h services.

The option ROM performs the following:

- The option ROM determines if it actually controls any Int 13h IPL devices.
- If there are no IPL devices, the option ROM simply returns from its BCV entry point. If there are IPL devices, the option ROM proceeds.
- The option ROM determines the number of hard drives currently installed by reading BDA address 0040:0075.
- If no other hard drives are installed (i.e., BDA 0040:0075 is zero, so this is the first hard drive) the current Int13h vector is copied to the Int 40h vector so that floppy services are handled properly.
- The option ROM infers the drive number for the current BCV IPL device from BDA 0040:0075 (i.e., BDA 0040:0075 + 80h).
- The option ROM hooks the Int13h vector, saving the current vector in order to chain to services for other drive numbers. Note that the option ROM may install Int 13h services for more than one drive in a single “hook”.
- The option ROM increments BDA 0040:0075 by the number of drives it installed services for, typically the number of drives installed is one.
- The option ROM returns from its entry point.
- At this point the option ROMs for all IPL devices have been initialized, all BCV device Int 13h services have been installed, and shadow RAM has been read-only enabled.
- The system BIOS sets the Int 18h vector to the address of its failed boot attempt recovery entry. This function is responsible for walking through the IPL priority list on failed boot attempts.
- The system BIOS sets the Int 19h vector to the address of its boot strap loader. This function is responsible for loading and invoking an IPL device’s bootstrap loader or BEV entry point.
- The system BIOS executes Int 19h. The first entry in the IPL priority list is the specified boot device.

Int 19h Processing:

- If the current IPL priority list selection is a BCV device:
- The system BIOS reads the device’s bootstrap loader from Sector 1, Head 0, Cylinder 0 to address 0000:7C00. The Int13h services for the BCV device’s associated drive number is used to read the loader.
- If a valid boot sector is detected, the system BIOS loads the OS bootstrap loader and calls its entry point. If the OS loader fails, it executes an Int 18h to indicate the failed boot. If the OS loader is successful, the OS is in control and there is no return to Int 19h.
- If the current IPL priority list selection is a BEV device:
- The system BIOS calls the BEV entry (extracted from the PnP expansion header). The BEV entry may also hook Int 13h, for example if the BEV is for an “El Torito” CD-ROM Hard Drive emulation.
- The BEV loads and executes its OS bootstrap loader.
- If the BEV or OS loader fails, it executes an Int 18h to indicate the failed boot. If the OS loader is successful, the OS is in control and there is no return to Int 19h.

- Int 18h is executed to indicate a failed boot.
- Int 18h Processing
- If all devices in the IPL priority list have been attempted:
- An error message/prompt is displayed indicating no OS was found.
- When the user responds to the prompt, the system BIOS executes Int 19h. The first entry in the IPL priority list is the specified boot device (i.e., starts over again).
- Otherwise, the system BIOS executes Int 19h, specifying the next relative IPL priority list entry as the boot device.

Single Option ROM Code Image

We now direct our attention to the description of the components and format of a single option ROM code image. Option ROMs that follow the Device Driver Initialization Model (DDIM) (PnP BIOS specification reference) exist in two forms:
Pre-initialization Image—the original option ROM image as programmed into the system-board or PnP card.
Post Initialization Image—the option ROM image as it exists in shadow RAM after its initialization entry has successfully returned.

Figures 2, and 3 show the logical components of an option ROM image which supports PnP, PCI, BBS, and DDIM. The individual elements are described below.

Figure 2: Pre-initialization Option ROM Image
Figure 3: Post-initialization Option ROM Image

Option ROM Components

PnP Option ROM Header: This is the root element of a Plug and Play option ROM. It identifies an image as an option ROM and provides links to the PCI Data Structure and PnP expansion header. The system BIOS uses the information in this header to:

- Detect a PnP option ROM image
- Copy (shadow) an image into RAM
- Validate an image via a checksum
- Initialize an image by calling its initialization entry
- Locate an image's PCI Data Structure
- Locate an image's PnP expansion header

Start with BIOS Boot Specification Version 1.01 for a detailed description.

PCI Data Structure: This is a companion item to the PnP option ROM Header. It qualifies an image as a PCI Expansion ROM and contains identification information for an image and its device.

Refer to PCI BIOS Specification Revision for a detailed description.

PnP Expansion Headers: This is a companion item to the PnP option ROM Header. In a pre-initialization image it identifies an image as a PnP option ROM. In a post-initialization image, one or more of these headers may exist, each identifying an IPL device, including:

- Reference to Product Name String
- Reference to BEV Code
- Reference to BCV Code
- Reference to the next PnP expansion header (i.e., linked list of IPL devices)

Start with BIOS Boot Specification Version 1.01 for a detailed description.

Runtime Code: The Runtime Code is the constant data and executable code that remains after an option ROM has successfully returned from its initialization entry. This code includes:

- BEV Code (BIOS Boot Specification Version 1.01)—code to directly load an OS from an IPL device and optionally hook Int 13h.
- BCV Code (BIOS Boot Specification Version 1.01)—code to detect an IPL device and optionally hook Int 13h.
- Int 13h Service Code—conventional and enhanced [EDD specification reference] Int 13h disk drive services.

Initialization Code: The Initialization Code is the constant data and executable code that is used only during initialization then discarded before returning from initialization entry. This code is responsible for:

- Device detection
- Device initialization
- Initializing runtime data structures
- Creation of Product Name String for each detected device
- Creation of PnP expansion header for each detected device
- "Disposing" of itself per the DDIM, see (Initialization Sequence paragraph reference)

Pad: Unused space used to expand an image to the next greatest 2048 byte boundary. Legacy requirements dictate that an image's size be a multiple of 2048 bytes.

Checksum: The checksum of the entire option ROM image. A value such that the 8-bit sum, using unsigned 2's complement addition ignoring overflow of all bytes including the checksum, is zero.

Product Name Strings: A Product Name String is a null terminated ASCII string that is intended to uniquely identify an IPL device. Currently only the first 32 characters are displayed and therefore significant.

Start with [BBS reference] for a detailed description.

Initialization Sequence: The Plug and Play (PnP) BIOS Specification (PnP BIOS Specification Reference) introduced the Device Driver Initialization Model for option ROMs. In this model, an option ROM image exists in two forms:

- Pre-initialization option ROM Image (Figure 1)—the image as it exists in ROM on the system board or PnP Card (Runtime Code and Initialization Code).
- Post-initialization option ROM Image (Figure 2)—the image as it exists in Shadow RAM after returning from its initialization entry (Runtime Code only).

The DDIM defines the transition from the Pre-initialization to the Post-initialization option ROM image. This transition facilitates:

- More efficient use of Shadow RAM by allowing initialization code (and data) to be discarded
- A means of dynamically building data structures and saving them as static (i.e., constant) data at boot time

Option ROM Initialization Sequence of Events:

- The system BIOS copies the ROM image into main memory.
- The system BIOS calls the image's initialization entry point, which is specified in the PnP option ROM Header. The option ROM performs the following:
- Initializes common data items.
- Searches for device(s) to control, for each device found:
- Initializes the device (reset, etc.) and its associated data structures.
- Creates a Product Name String to identify the device.

- Creates a PnP expansion header for the device and links it to the PnP expansion header List.
- If there are no devices to support:
- Writes zero to the Length field in its PnP option ROM Header.
- Returns from its initialization entry point with a status code indicating “no devices to support”.
- Or optionally, if the option ROM decides to leave a static footprint (An option ROM may want to leave an image resident in Shadow RAM to provide data to a related option ROM or driver, for example NVS data or Vital Product Data):
- Copies any data it wishes to be static to the end of the runtime image (i.e., overwrites code).
- Updates the Length field (Length = Pre-initialization Length – Code Length + Static Data Length) in its PnP option ROM Header. This effectively discards the Code and reduces the image’s footprint.
- Calculates the new image checksum and appends it to the new image.
- Returns from its initialization entry point with an appropriate status code.
- If there are devices to support, the option ROM will arrange and reduce its image as follows:
- Copies any data it wishes to be static to the end of the runtime image (i.e., overwrites Initialization Code). This would include the Product Name Strings and the PnP option ROM Headers.
- Updates the Length field (Length = Pre-initialization Length – Initialization Code Length + Static Data Length) in its PnP option ROM Header. This effectively discards the Initialization Code and reduces the image’s footprint.
- Calculates the new image checksum and appends it to the new image.
- Returns from its initialization entry point with a status code indicating “successful initialization”.

This article has attempted to document the legacy PC-AT boot process with special emphasis on the role of option ROMs. As part of the on-going industry effort to migrate away from legacy technologies and architectures, Intel intends to publish additional information that will describe a legacy-reduced IA server boot environment.

More Info

The following documents provide details on option ROMs and the PC-AT boot process, some of which are referenced in this article:

Sequence of Events:
IPL
PnP BBS System
Initialization

BIOS Boot Specification Version 1.01, Compaq Computer Corporation, Phoenix Technologies Ltd., Intel Corporation, 1996.

BIOS Enhanced Disk Drive Specification Version 3.0, Phoenix Technologies Ltd., 1998.

Clarification to Plug and Play BIOS Specification Version 1.0.

“El Torito” Bootable CD-ROM Format Specification Version 1.0, Phoenix Technologies, Ltd., IBM Corporation, 1994.

PCI BIOS Specification Revision (latest), PCI Special Interest Group, Hillsboro, OR.

Plug and Play BIOS Specification, Version 1.0A, Compaq Computer Corporation, Phoenix Technologies, Ltd., Intel Corporation, 1994, or ftp://download.intel.com/ial/wfm/bio10a.pdf.

POST Memory Manager Specification Version 1.01, Phoenix Technologies Ltd., Intel Corporation, 1997.

Author Bio

Roy Wade is a staff software engineer for LSI Logic Corporation. He is responsible for the design and development of BIOS and OS software to support LSI Logic's IO products. Roy has over 13 years of industry experience in a variety of areas, including satellite navigation, telecommunications, and Client/Server applications. Roy is a Phi Beta Kappa graduate of the University of North Dakota where he received his BS in Computer Science in 1986. Roy is a contributing member of the UNIX Developer's Interface Guide for IA-64 based servers (UDIG) working group. He has collaborated on patent applications in both the wireless communications and PC BIOS areas.

Ramin Neshati joined Intel in January 1998. Before becoming the developer guides technical program manager, he represented ESG on the Wired for Management initiative. Aside from Intel, Ramin has over 17 years of industry experience at S3 Incorporated, Dell Computer Corporation, Xerox Corporation and Link Systems Incorporated, in a variety of positions from software engineering to software architecture and engineering management. Ramin received his MBA from Pepperdine University (1993), the Master's degree in Computer Science from University of Idaho (1982) and the Bachelor's degree in Computer Science from Washington State University (1980). He has collaborated on several patent applications, published articles on networking protocols and services, and lectured at computer user's groups symposia, including SoftCon, EDGE and DECUS.